N88-16371

Software Simulation of Time Delay in Teleoperation

K. Wayne Goode
Kenneth E. Johnson Research Center
University of Alabama in Huntsville
Huntsville, Alabama 35899

ABSTRACT

This paper describes research done in the Space Robotics Laboratory at UAH studying the effects of time delay on teleoperation.

INTRODUCTION

The long range goal for the NASA space station is to establish a permanent presence of man and machines in space. Because of cost and safety factors, teleoperation will be important to the fulfillment of this goal.

Teleoperation means remote operation. That is, the robot receives instructions from a human operator and then performs the task. The task to be performed is at a remote distance from the operator.

In space applications there is often a delay between the time the human operator gives a command and time the command is executed in space. There is a further delay before pictures from cameras located at the remote site are relayed to the operator. As the time delay increases, so does the time required to complete the task, thus making the operation of the remote device more difficult.

The Johnson Research Center at the University of Alabama in Huntsville has established a space robotics laboratory to study time delay and other issues of teleoperation.

This paper will describe this lab and deal specifically with the software written to control the robot and simulate time delays.

HARDWARE

Figure 1 presents a system schematic of the space robotics laboratory. The laboratory is configured around a Puma 562 robot with 6 degrees of freedom (see Figure 2) which is on loan from Boeing Aerospace Company in Huntsville.

A custom designed joystick controller with two joysticks, each with three degrees of freedom, is used to control the robot. These joysticks are connected to the robot controller through an analog to digital interface.

SOFTWARE

The software to control the robot is written in VAL II, the control language for Puma robots.

Joystick Calibration

The joystick outputs a voltage in the range of 0 to 10 volts for each axis. The minimum, maximum, and center are different for each axis. The calibration program asks the operator to move each of the joysticks to its extreme positions. The program reads the minimum, maximum, and center voltages for each axis and stores the information for reference when reading the joystick.

The interface is subject to electrical noise. The voltages will vary by as much as 5%. However, since the joystick does not need to be very precise, this is an acceptable error. To reduce the error, the joystick is read several times and the values averaged.

Reading the Joystick

The subroutine JOYSTICK reads the joystick controller using the analog to digital interface. The reading are normalized in the range -1 to 1 based on the calibration information recorded by the calibration program. This number is squared to give more precise control around the center point.

Any reading less than .05 is considered to be 0. This dead zone around the center keeps the robot from drifting slightly due to small errors in the joystick reading caused by electrical noise.

Time Delays

The program DELAY (see listing 1) is the move program with a time delay added. The program prompts the user for the length of the time delay at the start of execution of the program.

When using a time delay, the program must read the joystick, store the information, recall other information and move the robot accordingly. The time required to execute each loop must be the same to keep the time delay accurate. The program executes this loop seven times a second. The steps it takes are:

1. Read the joystick.

2. Calculate the new position based on the readings.

3. If the position is out of range use the last valid position.

4. Store position on the top of stack.

5. Pull next position off the bottom of stack.

6. If the robot will complete the current motion by the end of the time allowed for the execution of the loop (1/7 of second), command the robot to move to the position just pulled off the stack.

7. Wait until the time allowed for execution of this loop has elapsed.

The robot can buffer one movement at time. That is, once a command is given to start a motion the program proceeds without waiting for the motion's completion. However, if another motion command is given before the first motion is finished, the program will wait for the first motion to finish and start the second motion before continuing with the program.

This can be a problem when the joystick must be continuously read. The solution is to skip the movement to the new location when the current movement will not be completed in the amount of time allowed. The robot will catch up during the next motion command.

The amount of time for each loop must be held constant so that the delay will be correct. An entry is read off the stack and a new one put on each time a loop is executed. This way, the time delay can be changed by varying the size of the stack. The stack should have seven entries for each second of delay because each step is 1/7 of a step long.

The computer's clock is incremented 35 times a second and the loop time must be a multiple of that time. 1/7 of a second was selected because it was the shortest time in which the steps necessary to each loop could be executed.

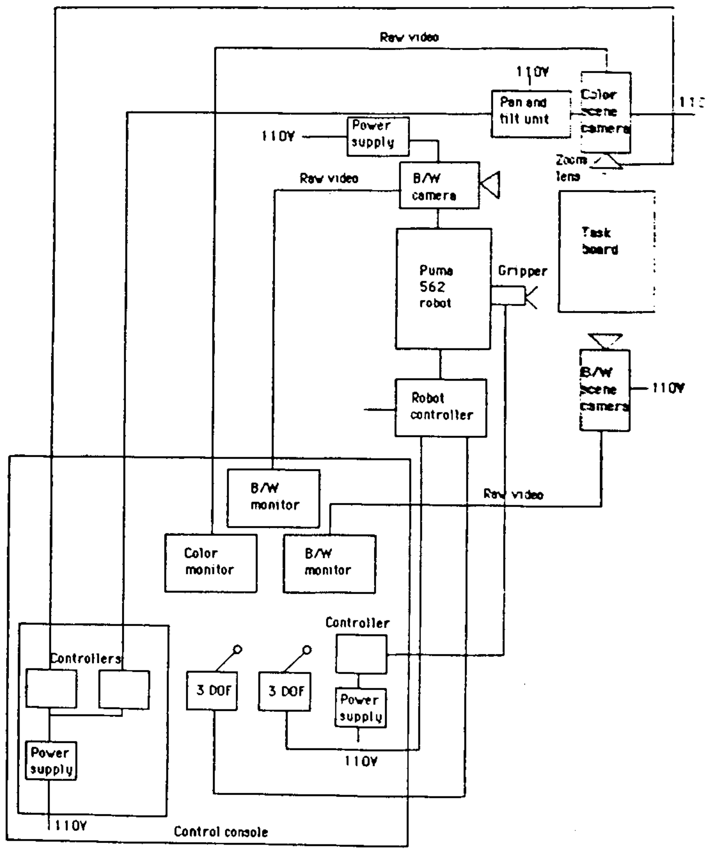

Figure 1 Space Automation and Robotics Laboratory

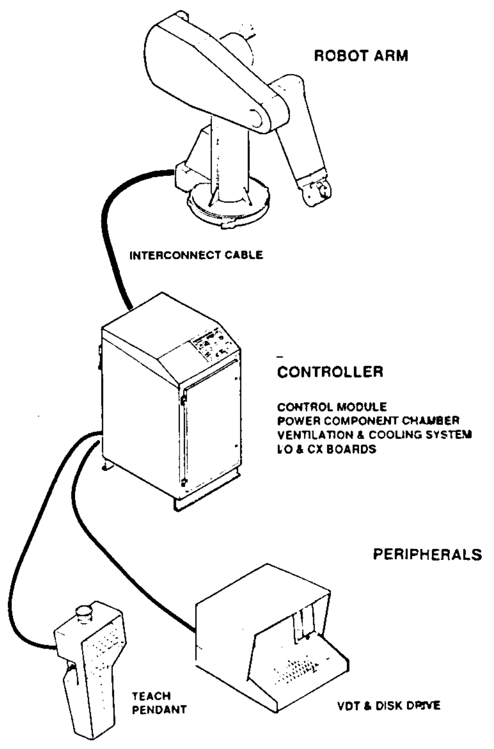

Figure 2 Puma 562 robot

VAL II ROBOT CONTROL LANGUAGE

The function of many of the command of the VAL II language are obvious. Some of the non-obvious commands and functions in the programs listings in Appendix B are explained below.

ADC	This function returns a value in the range -1023 to 1024 that represents a voltage in the range -10 to +10 for the analog to digital interface channel indicated.
BREAK	This command suspends program operation until the current robot motion has finished. Normally, the program starts a robot motion and continues with the program execution.
DECOMPOSE	This function returns the six components of a robot location. The components are x, y, z, orientation, altitude and rotation.
HERE	This command assigns the current location to the specified location variable.
INRANGE	This function returns a value indicating whether the specified location is in the robot's work envelope.
MOVE	This command moves the robot to the specified location.
RIGHTY	This commands sets the robot in a right handed configuration.
TIMER(-1)	This function returns the amount of time left before the current robot motion is finished.
TRANS	This function returns a location variable described by the six components given. This is the opposite of the function DECOMPOSE.
SET	This command is used to assign a value to a location variable.

Listing 1

DELAY PROGRAM

```
;Delay

;KW Goode 8JUL87

TYPE /C25, /U22, "Time delayed robot controlled program"
TYPE

CALL joystick
IF er THEN
    TYPE
    TYPE "The joystick power is turned off."
    RETURN
END

scale 1 = 20
```

```
     scale a = 5
     st = 0
     count = 0
     rate = 5
     tics = INT(tps + 0.5)

     RIGHTY
     MOVE TRANS (-800,0,-90,0,-90)
     TYPE "Moving the robot to the starting position"
     BREAK
     HERE rpos
     TYPE /U20,"
     TYPE /U20, /S
     PROMPT "Enter time delay in seconds ==>", delay
     TYPE
     steps = delay * rate
     IF steps < 1 THEN
     steps = 1
     END

     DECOMPOSE p[] = rpos
     FOR i = 0 to steps
         SET stack [i] = rpos
     END
     TIMER 1 = 0

10   SET dpos = stack [st]
     IF TIMER (-1) <tics/rate/TPS THEN
          MOVE dpos
     END
     CALL joystick
     p[0] = p[0] + x1*scalel
     p[1] = p[1] + y1*scalel
     p[2] = p[2] + z1*scalel
     p[3] = p[3] + x2*scalea
     p[4] = p[4] + y2*scalea
     p[5] = p[5] + z2*scalea
     SET rpos 1 = TRANS(p[0], p[1], p[2], p[3], p[4], p[5])
     IF INRANGE (rpos1) == 0 THEN
     SET rpos = rpos 1
     END
     DECOMPOSE p[] = rpos
     SET stack [st] = rpos
     st = st + 1
     IF st == steps THEN
        st = 0
     END
     count = count + 1
     WAIT TIMER (1) >= (count*tics/rate -0.5)/TPS
     GOTO 10
```